          FCAST: Scalable Object Delivery for the ALC and NORM Protocols
                        draft-ietf-rmt-fcast-04

Abstract

   This document introduces the FCAST object (e.g., file) delivery
   application on top of the ALC and NORM reliable multicast protocols.
   FCAST is a highly scalable application that provides a reliable
   object delivery service.

Status of this Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on January 28, 2012.

Copyright Notice

Table of Contents

1. Introduction

   This document introduces the FCAST reliable and scalable object
   (e.g., file) delivery application. Two variants of FCAST exist:

   o FCAST/ALC that relies on the Asynchronous Layer Coding (ALC)
      [RFC5775] and the Layered Coding Transport (LCT) [RFC5651]
      reliable multicast transport protocol, and

   o FCAST/NORM that relies on the NACK-Oriented Reliable Multicast
      (NORM) [RFC5740] reliable multicast transport protocol.

   Hereafter, the term FCAST denotes either FCAST/ALC or FCAST/NORM.
   FCAST is not a new protocol specification per se. Instead it is a
   set of data format specifications and instructions on how to use ALC
   and NORM to implement a file-casting service.

   Depending on the target use case, the delivery service provided by
   FCAST is more or less reliable. For instance, with FCAST/ALC used in
   ON-DEMAND mode over a time period that largely exceeds the typical
   download time, the service can be considered as fully reliable.
   Similarly, when FCAST is used along with a session control
   application that collects reception information and takes appropriate
   corrective measures (e.g., a direct point-to-point retransmission of
   missing packets, or a new multicast recovery session), then the
   service can be considered as fully reliable. On the opposite, if
   FCAST operates in PUSH mode, then the service is usually only
   partially reliable, and a receiver that is disconnected during a
   sufficient time will perhaps not have the possibility to download the
   object.

   Depending on the target use case, the FCAST scalability is more or
   less important. For instance, if FCAST/ALC is used on top of purely
   unidirectional transport channels, with no feedback information at
   all, which is the default mode of operation, then the scalability is
   maximum since neither FCAST, nor ALC, UDP or IP generates any
   feedback message. On the opposite, the FCAST/NORM scalability is
   typically limited by NORM scalability itself. Similarly, if FCAST is
   used along with a session control application that collects reception
   information from the receivers, then this session control application
   may limit the scalability of the global object delivery system. This
   situation can of course be mitigated by using a hierarchy of feedback
   message aggregators or servers. The details of this are out of the
   scope of the present document.

   A design goal behind FCAST is to define a streamlined solution, in
   order to enable lightweight implementations of the protocol stack,
   and limit the operational processing and storage requirements. A

consequence of this choice is that FCAST cannot be considered as a
versatile application, capable of addressing all the possible use-
cases. On the opposite, FCAST has some intrinsic limitations. From
this point of view it differs from FLUTE [RMT-FLUTE] which favors
flexibility at the expense of some additional complexity.

A good example of the design choices meant to favor the simplicity is
the way FCAST manages the object meta-data: by default, the meta-data
and the object content are sent together, in a compound object. This
solution has many advantages in terms of simplicity as will be
described later on. However, as such, it also has an intrinsic
limitation since it does not enable a receiver to decide in advance,
before beginning the reception of the compound object, whether the
object is of interest or not, based on the information that may be
provided in the meta-data. Therefore this document defines
additional techniques that may be used to mitigate this limitation.
It is also possible that some use-cases require that each receiver
download the whole set of objects sent in the session (e.g., with
mirroring tools). When this is the case, the above limitation is no
longer be a problem.

1.1. Applicability

FCAST is compatible with any congestion control protocol designed for
ALC/LCT or NORM. However, depending on the use-case, the data flow
generated by the FCAST application might not be constant, but instead
be bursty in nature. Similarly, depending on the use-case, an FCAST
session might be very short. Whether and how this will impact the
congestion control protocol is out of the scope of the present
document.

FCAST is compatible with any security mechanism designed for ALC/LCT
or NORM. The use of a security scheme is strongly RECOMMENDED (see
Section 6).

FCAST is compatible with any FEC scheme designed for ALC/LCT or NORM.
Whether FEC is used or not, and the kind of FEC scheme used, is to
some extent transparent to FCAST.

FCAST is compatible with both IPv4 and IPv6. Nothing in the FCAST
specification has any implication on the source or destination IP
address.

2. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in [RFC2119].

3. Definitions, Notations and Abbreviations

3.1. Definitions

   This document uses the following definitions:

   FCAST/ALC: denotes the FCAST application running on top of the ALC/
         LCT reliable transport protocol;

   FCAST/NORM: denotes the FCAST application running on top of the NORM
         reliable transport protocol;

   FCAST: denotes either FCAST/ALC or FCAST/NORM;

   Compound Object: denotes an ALC or NORM transport object composed of
         the Compound Object Header (Section 5.1), including any meta-
         data, and the content of the original application object
         (e.g., a file);

   Carousel: denotes the process of sending Compound Objects
         implemented by a FCAST sender;

   Carousel Instance: denotes a fixed set of registered Compound
         Objects that are sent by the carousel during a certain number
         of cycles. Whenever Compound Objects need to be added or
         removed, a new Carousel Instance is defined;

   Carousel Instance Descriptor (CID): denotes a special object that
         lists the Compound Objects that comprise a given Carousel
         Instance;

   Carousel Cycle: denotes a transmission round within which all the
         Compound Objects registered in the Carousel Instance are
         transmitted a certain number of times. By default, Compound
         Objects are transmitted once per cycle, but higher values are
         possible, that might differ on a per-object basis;

   Transmission Object Identifier (TOI): denotes the numeric identifier
         associated to a specific object by the underlying transport
         layer. In the case of ALC, this corresponds to the TOI
         described in that specification while for the NORM
         specification this corresponds to the NormTransportId
         described there.

3.2. Abbreviations

   This document uses the following abbreviations:

   CID: Carousel Instance Descriptor

   CIID: Carousel Instance IDentifier

   FEC OTI: FEC Object Transmission Information

   TOI: Transmission Object Identifier

4. FCAST Principles

4.1. FCAST Content Delivery Service

   The basic goal of FCAST is to transmit objects to a group of
   receivers in a reliable way. The receiver set MAY be restricted to a
   single receiver or MAY include possibly a very large number of
   receivers. FCAST is specified to support two forms of operation:

   1. FCAST/ALC: where the FCAST application is meant to run on top of
       the ALC/LCT reliable multicast transport protocol, and

   2. FCAST/NORM: where the FCAST application is meant to run on top of
       the NORM reliable multicast transport protocol.

   This specification is designed such that both forms of operation
   share as much commonality as possible.

   While the choice of the underlying transport protocol (i.e., ALC or
   NORM) and its parameters may limit the practical receiver group size,
   nothing in FCAST itself limits it. The transmission might be fully
   reliable, or only partially reliable depending upon the way ALC or
   NORM is used (e.g., whether FEC encoding and/or NACK-based repair
   requests are used or not), the way the FCAST carousel is used (e.g.,
   whether the objects are made available for a long time span or not),
   and the way in which FCAST itself is employed (e.g., whether there is
   a session control application that might automatically extend an
   existing FCAST session until all receivers have received the
   transmitted content).

   FCAST is designed to be as self-sufficient as possible, in particular
   in the way object meta-data is attached to object data content.
   However, for some uses, meta-data MAY also be communicated by an out-
   of-band mechanism that is out of the scope of the present document.

4.2. Meta-Data Transmission

   FCAST usually carries meta-data elements by prepending them to the
   object it refers to. As a result, a Compound Object is created that
   is composed of a header followed by the original object data. This
   header is itself composed of the meta-data as well as several fields,
   for instance to indicate the boundaries between the various parts of
   this Compound Object (Figure 1).

```
   <----------------------- Compound Object ----------------------->
   +------------------------+--------------------------------------+
   | Compound Object Header  |              Object Data              |
   | (can include meta-data) |        (can be encoded by FCAST)      |
   +------------------------+--------------------------------------+
```
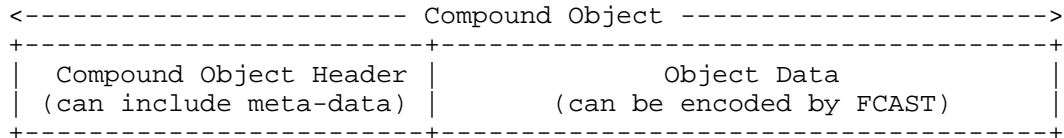
                 Figure 1: Compound Object composition.

   Attaching the meta-data to the object is an efficient solution, since
   it guaranties that meta-data be received along with the associated
   object, and it allows the transport of the meta-data to benefit from
   any transport-layer erasure protection of the Compound Object (e.g.,
   using FEC encoding and/or NACK-based repair). However a limit of
   this scheme, as such, is that a client does not know the meta-data of
   an object before beginning its reception, and in case of erasures
   affecting the meta-data, not until the object decoding is completed.
   The details of course depend upon the transport protocol and the FEC
   code used.

   In certain use-cases, FCAST can also be associated to another in-band
   (e.g., via NORM INFO messages, Section 4.8) or out-of-band signaling
   mechanism. In that case, this mechanism can be used in order to
   carry the whole meta-data (or a subset of it), possibly ahead of
   time.

4.3. Meta-Data Content

   The meta-data associated to an object can be composed of, but are not
   limited to:

   o Content-Location: the URI of the object, which gives the name and
      location of the object;

   o Content-Type: the MIME type of the object;

   o Content-Length: the size of the initial object, before any content
      encoding (if any). Note that this content length does not include
      the meta-data nor the fixed size Compound Object header;

o Content-Encoding: the optional encoding of the object performed by
   FCAST. If there is no Content-Encoding entry, the receiver MUST
   assume that the object is not encoded (default). The support of
   gzip encoding, or any other solution, remains optional.

o Content-MD5: the MD5 message digest of the object in order to
   check its integrity. Note that this digest is meant to protect
   from transmission and processing errors, not from deliberate
   attacks by an intelligent attacker. Note also that this digest
   only protects the object, not the header, and therefore not the
   meta-data. A separate checksum is provided to that purpose
   (Section 5.1);

o a digital signature for this object;

This list is not limited and new meta-data information can be added.
For instance, when dealing with very large objects (e.g., that
largely exceed the working memory of a receiver), it can be
interesting to split this object into several sub-objects (or
slices). When this happens, the meta-data associated to each sub-
object MUST include the following entries:

o Fcast-Obj-Slice-Nb: the total number of slices. A value strictly
   greater than 1 indicates that this object is the result of a split
   of the original object;

o Fcast-Obj-Slice-Idx: the slice index (in the {0 .. SliceNb - 1}
   interval);

o Fcast-Obj-Slice-Offset: the offset at which this slice starts
   within the original object;

When meta-data elements are communicated out-of-band, in advance of
data transmission, the following pieces of information MAY also be
useful:

o TOI: the Transmission Object Identifier (TOI) of the object
   (Section 4.6), in order to enable a receiver to easily associate
   the meta-data to the object;

o FEC Object Transmission Information (FEC OTI). In this case the
   FCAST sender does not need to use the optional EXT_FTI mechanism
   of the ALC or NORM protocols.

4.4. Carousel Transmission

   A set of FCAST Compound Objects scheduled for transmission are
   considered a logical "Carousel". A given "Carousel Instance" is
   comprised of a fixed set of Compound Objects. Whenever the FCAST
   application needs to add new Compound Objects to, or remove old
   Compound Objects from the transmission set, a new Carousel Instance
   is defined since the set of Compound Objects changes. Because of the
   native object multiplexing capability of both ALC and NORM, sender
   and receiver(s) are both capable to multiplex and demultiplex FCAST
   Compound Objects.

   For a given Carousel Instance, one or more transmission cycles are
   possible. During each cycle, all of the Compound Objects comprising
   the Carousel are sent. By default, each object is transmitted once
   per cycle. However, in order to allow different levels of priority,
   some objects MAY be transmitted more often that others during a
   cycle, and/or benefit from higher FEC protection than others. This
   can be the case for instance for the CID objects (Section 4.5). For
   some FCAST usage (e.g., a unidirectional "push" mode), a Carousel
   Instance may be associated to a single transmission cycle. In other
   cases it may be associated to a large number of transmission cycles
   (e.g., in "on-demand" mode, where objects are made available for
   download during a long period of time).

4.5. Carousel Instance Descriptor Special Object

   The FCAST sender CAN transmit an OPTIONAL Carousel Instance
   Descriptor (CID). The CID carries the list of the Compound Objects
   that are part of a given Carousel Instance, by specifying their
   respective Transmission Object Identifiers (TOI). However the CID
   does not describe the objects themselves (i.e., there is no meta-
   data). Additionally, the CID MAY include a "Complete" flag that is
   used to indicate that no further modification to the enclosed list
   will be done in the future. Finally, the CID MAY include a Carousel
   Instance ID that identifies the Carousel Instance it pertains to.
   These aspects are discussed in Section 5.2.

   There is no reserved TOI value for the CID Compound Object itself,
   since this special object is regarded by ALC/LCT or NORM as a
   standard object. On the opposite, the nature of this object (CID) is
   indicated by means of a specific Compound Object header field (the
   "I" flag) so that it can be recognized and processed by the FCAST
   application as needed. A direct consequence is the following: since
   a receiver does not know in advance which TOI will be used for the
   following CID (in case of a dynamic session), he MUST NOT filter out
   packets that are not in the current CID's TOI list. Said
   differently, the goal of CID is not to setup ALC or NORM packet

filters (this mechanism would not be secure in any case).

The use of a CID remains optional. If it is not used, then the
clients progressively learn what files are part of the carousel
instance by receiving ALC or NORM packets with new TOIs. However
using a CID has several benefits:

o When the "Complete" flag is set (if ever), the receivers know when
   they can leave the session, i.e., when they have received all the
   objects that are part of the last carousel instance of this
   delivery session;

o In case of a session with a dynamic set of objects, the sender can
   reliably inform the receivers that some objects have been removed
   from the carousel with the CID. This solution is more robust than
   the "Close Object flag (B)" of ALC/LCT since a client with an
   intermittent connectivity might loose all the packets containing
   this B flag. And while NORM provides a robust object cancellation
   mechanism in the form of its NORM_CMD(SQUELCH) message in response
   to receiver NACK repair requests, the use of the CID provides an
   additional means for receivers to learn of objects for which it is
   futile to request repair;

o The TOI equivalence (Section 4.6) can be signaled with the CID.
   This is often preferable to the alternative solution where the
   equivalence is identified by examining the object meta-data,
   especially in case of erasures.

During idle periods, when the carousel instance does not contain any
object, a CID with an empty TOI list MAY be transmitted. In that
case, a new carousel instance ID MUST be used to differentiate this
(empty) carousel instance from the other ones. This mechanism can be
useful to inform the receivers that:

o all the previously sent objects have been removed from the
   carousel. It therefore improves the FCAST robustness even during
   "idle" period;

o the session is still active even if there is currently no content
   being sent. Said differently, it can be used as a heartbeat
   mechanism. If the "Complete" flag has not been set, it implicitly
   informs the receivers that new objects MAY be sent in the future;

The decisions of whether a CID should be used or not, how often and
when a CID should be sent, are left to the sender and depend on many
parameters, including the target use case and the session dynamics.
For instance it may be appropriate to send a CID at the beginning of
each new carousel instance, and then periodically. These operational

aspects are out of the scope of the present document.

4.6. Compound Object Identification

The FCAST Compound Objects are directly associated with the object-
based transport service that the ALC and NORM protocols provide. In
each of these protocols, the messages containing transport object
content are labeled with a numeric transport object identifier (i.e.,
the ALC TOI and the NORM NormTransportId). For the purposes of this
document, this identifier in either case (ALC or NORM) is referred to
as the TOI.

There are several differences between ALC and NORM:

o the ALC use of TOI is rather flexible, since several TOI field
   sizes are possible (from 16 to 112 bits), since this size can be
   changed at any time, on a per-packet basis, and since the TOI
   management is totally free as long as each object is associated to
   a unique TOI (if no wraparound happened);

o the NORM use of TOI is more directive, since the TOI field is 16
   bit long and since TOIs MUST be managed sequentially;

In both NORM and ALC, it is possible that the transport
identification space may eventually wrap for long-lived sessions
(especially with NORM where this phenomenon is expected to happen
more frequently). This can possibly introduce some ambiguity in
FCAST object identification if a sender retains some older objects in
newer Carousel Instances with updated object sets. To avoid
ambiguity the active TOIs (i.e., the TOIs corresponding to objects
being transmitted) can only occupy half of the TOI sequence space.
If an old object, whose TOI has fallen outside the current window,
needs to be transmitted again, a new TOI must be used for it. In
case of NORM, this constraint limits to 32768 the maximum number of
objects that can be part of any carousel instance. In order to allow
receivers to properly combine the transport packets with a newly-
assigned TOI to those of associated to the previously-assigned TOI, a
mechanism is required to equate the objects with the new and the old
TOIs.

The preferred mechanism consists in signaling, within the CID, that
the newly assigned TOI, for the current Carousel Instance, is
equivalent to the TOI used within a previous Carousel Instance. By
convention, the reference tuple for any object is the {TOI; CI ID}
tuple used for its first transmission within a Carousel Instance.
This tuple MUST be used whenever a TOI equivalence is provided.

An alternative solution, when meta-data can be processed rapidly

(e.g., by using NORM-INFO messages), consists for the receiver in
identifying that both objects are the same, after examining the meta-
data. The receiver can then take appropriate measures.

4.7. FCAST/ALC Additional Specificities

There are no additional detail or option for FCAST/ALC operation.

4.8. FCAST/NORM Additional Specificities

The NORM Protocol provides a few additional capabilities that can be
used to specifically support FCAST operation:

1. The NORM_INFO message can convey "out-of-band" content with
    respect to a given transport object. With FCAST, it MAY be used
    to provide to the receivers a new, associated, Compound Object
    which contains the main Compound Object meta-data, or a subset of
    it. In that case the NORM_INFO Compound Object MUST NOT contain
    any Object Data field (i.e., it is only composed of the header),
    it MUST feature a non global checksum, and it MUST NOT include
    any padding field. The main Compound Object MUST in any case
    contain the whole meta-data (e.g., because a receiver MAY not
    support the NORM_INFO facility). Additionally, the meta-data
    entries contained in the NORM_INFO MUST be identical to the same
    entries in the main Compound Object. Finally, note that the
    availability of NORM_INFO for a given object is signaled through
    the use of a dedicated flag in the NORM_DATA message header.
    Along with NORM's NACK-based repair request signaling, it allows
    a receiver to quickly (and independently) request an object's
    NORM_INFO content. However, a limitation here is that the
    NORM_INFO Compound Object header MUST fit within the byte size
    limit defined by the NORM sender's configured "segment size"
    (typically a little less than the network MTU);

2. The NORM_CMD(SQUELCH) messages are used by the NORM protocol
    sender to inform receivers of objects that have been canceled
    when receivers make repair requests for such invalid objects.
    Along with the CID mechanism, a receiver has two efficient and
    reliable ways to discover old objects that have been removed from
    the carousel instance;

3. NORM also supports an optional positive acknowledgment mechanism
    that can be used for small-scale multicast receiver group sizes.
    Also, it may be possible in some cases for the sender to infer,
    after some period without receiving NACKs at the end of its
    transmission that the receiver set has fully received the
    transmitted content. In particular, if the sender completes its
    end-of-transmission series of NORM_CMD(FLUSH) messages without

receiving repair requests from the group, it may have some
assurance that the receiver set has received the content prior to
that point. These mechanisms are likely to help FCAST in
achieving fully reliable transmissions;

It should be noted that the NORM_INFO message header may carry the
EXT_FTI extension. The reliable delivery of the NORM_INFO content
allows the individual objects' FEC Transmission Information to be
provided to the receivers without burdening every packet (i.e.
NORM_DATA messages) with this additional, but important, content.
Examples are provided in Appendix A.

4.9. FCAST Sender Behavior

The following operations MAY take place at a sender:

1. The user (or another application) selects a set of objects (e.g.,
    files) to deliver and submits them, along with their meta-data,
    to the FCAST application;

2. For each object, FCAST creates the Compound Object and registers
    this latter in the carousel instance;

3. The user then informs FCAST that all the objects of the set have
    been submitted. If the user knows that no new object will be
    submitted in the future (i.e., if the session's content is now
    complete), the user informs FCAST. Finally, the user specifies
    how many transmission cycles are desired (this number may be
    infinite);

4. At this point, the FCAST application knows the full list of
    Compound Objects that are part of the Carousel Instance and can
    create a CID if desired, possibly with the complete flag set;

5. The FCAST application can now define a transmission schedule of
    these Compound Objects, including the optional CID. This
    schedule defines in which order the packets of the various
    Compound Objects should be sent. This document does not specify
    any scheme. This is left to the developer within the provisions
    of the underlying ALC or NORM protocol used and the knowledge of
    the target use-case.

6. The FCAST application then starts the carousel transmission, for
    the number of cycles specified. Transmissions take place until:

    * the desired number of transmission cycles has been reached, or

   * the user wants to prematurely stop the transmissions, or

   * the user wants to add one or several new objects to the
      carousel, or on the opposite wants to remove old objects from
      the carousel. In that case a new carousel instance must be
      created.

   7. If the session is not finished, then continue at Step 1 above;

4.10. FCAST Receiver Behavior

   The following operations MAY take place at a receiver:

   1. The receiver joins the session and collects incoming packets;

   2. If the header portion of a Compound Object is entirely received
       (which may happen before receiving the entire object with some
       ALC/NORM configurations), or if the meta-data is sent by means of
       another mechanism prior to the object, the receiver processes the
       meta-data and chooses to continue to receive the object content
       or not;

   3. When a Compound Object has been entirely received, the receiver
       processes the header, retrieves the object meta-data, perhaps
       decodes the meta-data, and processes the object accordingly;

   4. When a CID is received, which is indicated by the 'C' flag set in
       the Compound Object header, the receiver decodes the CID, and
       retrieves the list of objects that are part of the current
       carousel instance. This list CAN be used to remove objects sent
       in a previous carousel instance that might not have been totally
       decoded and that are no longer part of the current carousel
       instance;

   5. When a CID is received, the receiver also retrieves the list of
       TOI equivalences, if any, and takes appropriate measures, for
       instance by informing the transport layer;

   6. When a receiver has received a CID with the "Complete" flag set,
       and has successfully received all the objects of the current
       carousel instance, it can safely exit from the current FCAST
       session;

   7. Otherwise continue at Step 2 above.

5. FCAST Data Formats

    This section details the various data formats used by FCAST.

5.1. Compound Object Header Format

    In an FCAST session, Compound Objects are constructed by prepending
    the Compound Object Header (which may include meta-data) before the
    original object data content (Figure 2).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ^
|Ver| Resvd |G|C| MDFmt | MDEnc |            Checksum           |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                 Compound Object Header Length                 |  h
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  d
|            Object Meta-Data (optional, variable length)       |  r
|                         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+      |  |
|                         |          Padding (optional)         |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  v
|                                                               |
.             Object Data (optional, variable length)          .
.                                                               .
.                                                               .
```
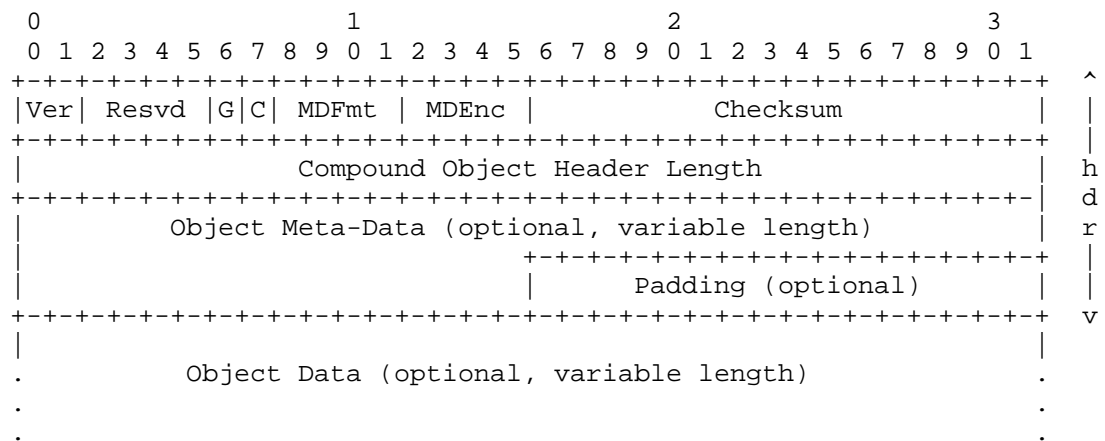
                Figure 2: Compound Object Header with Meta-Data.

    The Compound Object Header fields are:

Field	Description
Version	2-bit field that MUST be set to 0 in this specification and indicates the protocol version number.
Reserved	4-bit field that MUST be set to 0 in this specification and is reserved for future use. Receivers MUST ignore this field.
G	1-bit field that, when set to 1, indicates that the checksum encompasses the whole Compound Object (Global checksum). When set to 0, this field indicates that the checksum encompasses only the Compound Object header.
C	1-bit field that, when set to 1, indicates the object is a Carousel Instance Descriptor (CID). When set to 0, this field indicates that the transported object is a standard object.

Meta-Data Format (MDFmt)	4-bit field that defines the format of the object meta-data (see Section 7). An HTTP/1.1 metainformation format [RFC2616] MUST be supported and is associated to value 0. Other formats (e.g., XML) MAY be defined in the future.
Meta-Data Encoding (MDEnc)	4-bit field that defines the optional encoding of the Object Meta-Data field (see Section 7). By default, a plain text encoding is used and is associated to value 0. Gzip encoding MUST also be supported and is associated to value 1. Other encodings MAY be defined in the future.
Checksum	16-bit field that contains the checksum computed over either the whole Compound Object (when G is set to 1), or over the Compound Object header (when G is set to 0), using the Internet checksum algorithm specified in [RFC1071]. More precisely, the checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words to be considered. If a segment contains an odd number of octets to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes (this pad is not transmitted). While computing the checksum, the checksum field itself is set to zero.
Compound Object Header Length	32-bit field indicating total length (in bytes) of all fields of the Compound Object Header, except the optional padding. A header length field set to value 8 means that there is no meta-data included. When this size is not multiple to 32-bits words and when the Compound Object Header is followed by a non null Compound Object Data, padding MUST be added. It should be noted that the meta-data field maximum size is equal to $(2^{32} - 8)$ bytes.
Object Meta-Data	Optional, variable length field that contains the meta-data associated to the object. The format and encoding of this field is defined by the MDFmt MDEnc fields. With the default HTTP/1.1 format and plain text encoding, the Meta-Data is NULL-terminated plain text that follows the "TYPE" ":" "VALUE" "<CR-LF>" format used in HTTP/1.1 for metainformation [RFC2616]. The various meta-data items can appear in any order. The associated string, when non empty, MUST be NULL-terminated. When no meta-data is communicated, this field MUST be empty and the Compound Object Header Length MUST be equal to 8.

```
|   Padding  | Optional, variable length field of zero-value bytes |
|            | to align the start of the Object Data to 32-bit     |
|            | boundary.  Padding is only used when the Compound   |
|            | Object Header Length value, in bytes, is not        |
|            | multiple of 4 and when the Compound Object Header is |
|            | followed by non null Compound Object Data.          |
+-----------+-----------------------------------------------------+
```

The Compound Object Header is then followed by the Object Data, i.e.,
the original object possibly encoded by FCAST. Note that the length
of this content is the transported object length (e.g., as specified
by the FEC OTI) minus the Compound Object Header Length and optional
padding if any.

5.2. Carousel Instance Descriptor Format

   The format of the CID, which is a special Compound Object, is given
   in Figure 3.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    ^
    |Ver| Resvd |G|C| MDFmt | MDEnc |            Checksum           |    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    |
    |                 Compound Object Header Length                 |    h
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|    d
    |         Object Meta-Data (optional, variable length)          |    r
    |                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
    |                      |           Padding (optional)        |  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    v
    .                                                             .    ^
    .             Object List (variable length)                  .    |
    .                                                             .    o
    .                                    +-+-+-+-+-+-+-+-+-+       .    b
    .                                    |                        .    j
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                             v
```
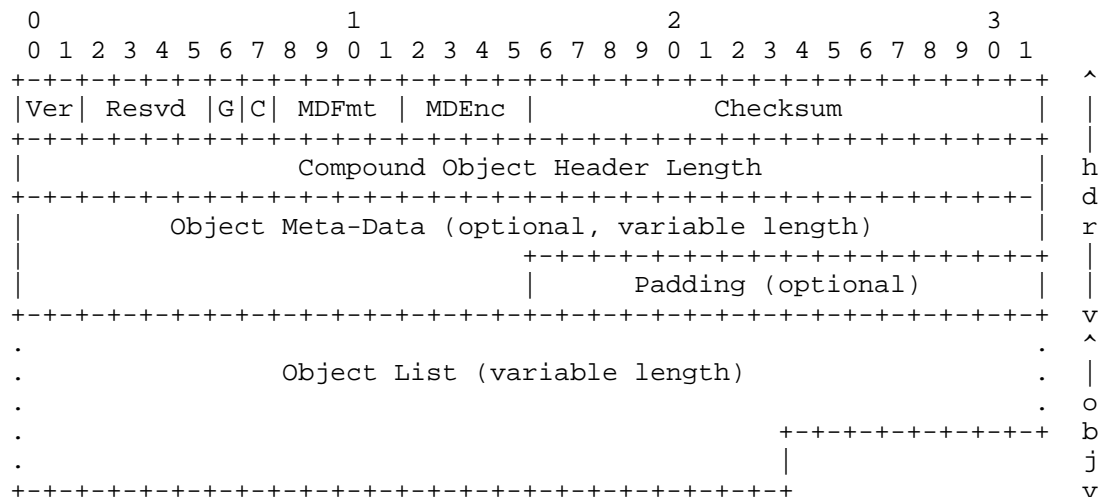
                  Figure 3: Carousel Instance Object Format.

   Because the CID is transmitted as a special Compound Object, the
   following CID-specific meta-data entries are defined:

   o Fcast-CID-Complete: when set to 1, it indicates that no new
      objects in addition to the ones whose TOI are specified in this
      CID, or the ones that have been specified in the previous CID(s),
      will be sent in the future. Otherwise it MUST be set to 0. This
      entry is optional. If absent, a receiver MUST conclude that the
      session is complete.

o Fcast-CID-ID: this entry contains the Carousel Instance
   IDentifier, or CIID. It starts from 0 and is incremented by 1 for
   each new carousel instance. This entry is optional if the FCAST
   session consists of a single, complete, carousel instance. In all
   other cases, this entry MUST be defined. In particular, the CIID
   is used by the TOI equivalence mechanism thanks to which any
   object is uniquely identified, even if the TOI is updated (e.g.,
   after re-enqueuing the object with NORM). The Fcast-CID-ID value
   can also be useful to detect possible gaps in the Carousel
   Instances, for instance caused by long disconnection periods.
   Finally, it can also be useful to avoid problems when TOI wrapping
   to 0 takes place to differentiate the various incarnations of the
   TOIs if need be.

The motivation for making the Fcast-CID-Complete and Fcast-CID-ID
entries optional is to simplify the simple case of a session
consisting of a single, complete, carousel instance, with an Object
List given in plain text, without any content encoding. In that
case, the CID does not need to contain any meta-data entry.

Additionally, the following standard meta-data entries are often used
(Section 4.3):

o Content-Length: it specifies the size of the object list, before
   any content encoding (if any).

o Content-Encoding: it specifies the optional encoding of the object
   list, performed by FCAST. For instance:
        Content-Encoding: gzip
   indicates that the Object List field has been encoded with gzip
   [RFC1952]. If there is no Content-Encoding entry, the receiver
   MUST assume that the Object List field is plain text (default).
   The support of gzip encoding, or any other solution, remains
   optional.

An empty Object List is valid and indicates that the current carousel
instance does not include any object (Section 4.5). This can be
specified by using the following meta-data entry:
        Content-Length: 0
or simply by leaving the Object List empty. In both cases, padding
MUST NOT be used and consequently the transported object length
(e.g., as specified by the FEC OTI) minus the Compound Object Header
Length equals zero.

The non-encoded (i.e., plain text) Object List, when non empty, is a
NULL-terminated ASCII string. It can contain two things:

o a list of TOI values, and

o a list of TOI equivalences;

First of all, this string can contain the list of TOIs included in
the current carousel instance, specified either as the individual
TOIs of each object, or as TOI intervals, or any combination. The
format of the ASCII string is a comma-separated list of individual
"TOI" values or "TOI_a-TOI_b" elements. This latter case means that
all values between TOI_a and TOI_b, inclusive, are part of the list.
In that case TOI_a MUST be strictly inferior to TOI_b. If a TOI
wrapping to 0 occurs in an interval, then two TOI intervals MUST be
specified, TOI_a-MAX_TOI and 0-TOI_b.

This string can also contain the TOI equivalences, if any. The
format is a comma-separated list of "(" newTOI "=" 1stTOI "/" 1stCIID
")" elements. Each element says that the new TOI, for the current
Carousel Instance, is equivalent to (i.e., refers to the same object
as) the provided identifier, 1stTOI, for the Carousel Instance of ID
1stCIID.

The ABNF specification is the following:
```
cid-list   =  *(list-elem *( "," list-elem))
list-elem    =  toi-elem / toieq-elem
toi-elem    =  toi-value / toi-interval
toi-value   =  1*DIGIT
toi-interval =  toi-value "-" toi-value
               ; additionally, the first toi-value MUST be
               ; strictly inferior to the second toi-value
toieq-elem   =  "(" toi-value "=" toi-value "/" ciid-value ")"
ciid-value   =  1*DIGIT
DIGIT       =  %x30-39
               ; a digit between 0 and 9, inclusive
```

For readability purposes, it is RECOMMENDED that all the TOI values
in the list be given in increasing order. However a receiver MUST be
able to handle non-monotonically increasing values. It is also
RECOMMENDED to group the TOI equivalence elements together, at the
end of the list, in increasing newTOI order. However a receiver MUST
be able to handle lists of mixed TOI and TOI equivalence elements.
Specifying a TOI equivalence for a given newTOI relieves the sender
from specifying newTOI explicitly in the TOI list. However a
receiver MUST be able to handle situations where the same TOI appears
both in the TOI value and TOI equivalence lists. Finally, a given
TOI value or TOI equivalence item MUST NOT be included multiple times
in either list.

For instance, the following object list specifies that the current

Carousel Instance is composed of 8 objects, and that TOIs 100 to 104
are equivalent to the TOIs 10 to 14 of Carousel Instance ID 2 and
refer to the same objects:

97,98,99,(100=10/2),(101=11/2),(102=12/2),(103=13/2),(104=14/2)

or equivalently:

97-104,(100=10/2),(101=11/2),(102=12/2),(103=13/2),(104=14/2)

6. Security Considerations

6.1. Problem Statement

A content delivery system is potentially subject to attacks. Attacks
may target:

o the network (to compromise the routing infrastructure, e.g., by
   creating congestion),

o the Content Delivery Protocol (CDP) (e.g., to compromise the
   normal behavior of FCAST), or

o the content itself (e.g., to corrupt the objects being
   transmitted).

These attacks can be launched either:

o against the data flow itself (e.g., by sending forged packets),

o against the session control parameters (e.g., by corrupting the
   session description, the CID, the object meta-data, or the ALC/LCT
   control parameters), that are sent either in-band or out-of-band,
   or

o against some associated building blocks (e.g., the congestion
   control component).

In the following sections we provide more details on these possible
attacks and sketch some possible counter-measures. We finally
provide recommendations in Section 6.5.

6.2. Attacks Against the Data Flow

Let us consider attacks against the data flow first. At least, the
following types of attacks exist:

   o attacks that are meant to give access to a confidential object
      (e.g., in case of a non-free content) and

   o attacks that try to corrupt the object being transmitted (e.g., to
      inject malicious code within an object, or to prevent a receiver
      from using an object, which is a kind of Denial of Service (DoS)).

6.2.1. Access to Confidential Objects

   Access control to the object being transmitted is typically provided
   by means of encryption. This encryption can be done over the whole
   object (e.g., by the content provider, before submitting the object
   to FCAST), or be done on a packet per packet basis (e.g., when IPsec/
   ESP is used [RFC4303], see Section 6.5). If confidentiality is a
   concern, it is RECOMMENDED that one of these solutions be used.

6.2.2. Object Corruption

   Protection against corruptions (e.g., if an attacker sends forged
   packets) is achieved by means of a content integrity verification/
   sender authentication scheme. This service can be provided at the
   object level, but in that case a receiver has no way to identify
   which symbol(s) is(are) corrupted if the object is detected as
   corrupted. This service can also be provided at the packet level.
   In this case, after removing all corrupted packets, the file may be
   in some cases recovered. Several techniques can provide this content
   integrity/sender authentication service:

   o at the object level, the object can be digitally signed, for
      instance by using RSASSA-PKCS1-v1_5 [RFC3447]. This signature
      enables a receiver to check the object integrity, once this latter
      has been fully decoded. Even if digital signatures are
      computationally expensive, this calculation occurs only once per
      object, which is usually acceptable;

   o at the packet level, each packet can be digitally signed
      [RMT-SIMPLE-AUTH]. A major limitation is the high computational
      and transmission overheads that this solution requires. To avoid
      this problem, the signature may span a set of packets (instead of
      a single one) in order to amortize the signature calculation. But
      if a single packets is missing, the integrity of the whole set
      cannot be checked;

   o at the packet level, a Group Message Authentication Code (MAC)
      [RFC2104][RMT-SIMPLE-AUTH] scheme can be used, for instance by
      using HMAC-SHA-256 with a secret key shared by all the group
      members, senders and receivers. This technique creates a
      cryptographically secured digest of a packet that is sent along

with the packet. The Group MAC scheme does not create prohibitive
processing load nor transmission overhead, but it has a major
limitation: it only provides a group authentication/integrity
service since all group members share the same secret group key,
which means that each member can send a forged packet. It is
therefore restricted to situations where group members are fully
trusted (or in association with another technique as a pre-check);

o at the packet level, Timed Efficient Stream Loss-Tolerant
   Authentication (TESLA) [RFC4082][RFC5776] is an attractive
   solution that is robust to losses, provides a true authentication/
   integrity service, and does not create any prohibitive processing
   load or transmission overhead. Yet checking a packet requires a
   small delay (a second or more) after its reception;

o at the packet level, IPsec/ESP [RFC4303] can be used to check the
   integrity and authenticate the sender of all the packets being
   exchanged in a session (see Section 6.5).

Techniques relying on public key cryptography (digital signatures and
TESLA during the bootstrap process, when used) require that public
keys be securely associated to the entities. This can be achieved by
a Public Key Infrastructure (PKI), or by a PGP Web of Trust, or by
pre-distributing securely the public keys of each group member.

Techniques relying on symmetric key cryptography (Group MAC) require
that a secret key be shared by all group members. This can be
achieved by means of a group key management protocol, or simply by
pre-distributing securely the secret key (but this manual solution
has many limitations).

It is up to the developer and deployer, who know the security
requirements and features of the target application area, to define
which solution is the most appropriate. In any case, whenever there
is any concern of the threat of file corruption, it is RECOMMENDED
that at least one of these techniques be used.

6.3. Attacks Against the Session Control Parameters and Associated
      Building Blocks

   Let us now consider attacks against the session control parameters
   and the associated building blocks. The attacker has at least the
   following opportunities to launch an attack:

   o the attack can target the session description,

   o the attack can target the FCAST CID,

o   the attack can target the meta-data of an object,

o   the attack can target the ALC/LCT parameters, carried within the
    LCT header or

o   the attack can target the FCAST associated building blocks, for
    instance the multiple rate congestion control protocol.

The consequences of these attacks are potentially serious, since they
can compromise the behavior of content delivery system or even
compromise the network itself.

6.3.1. Attacks Against the Session Description

An FCAST receiver may potentially obtain an incorrect Session
Description for the session. The consequence of this is that
legitimate receivers with the wrong Session Description are unable to
correctly receive the session content, or that receivers
inadvertently try to receive at a much higher rate than they are
capable of, thereby possibly disrupting other traffic in the network.

To avoid these problems, it is RECOMMENDED that measures be taken to
prevent receivers from accepting incorrect Session Descriptions. One
such measure is the sender authentication to ensure that receivers
only accept legitimate Session Descriptions from authorized senders.
How these measures are achieved is outside the scope of this document
since this session description is usually carried out-of-band.

6.3.2. Attacks Against the FCAST CID

Corrupting the FCAST CID is one way to create a Denial of Service
attack. For example, the attacker can set the "Complete" flag to
make the receivers believe that no further modification will be done.

It is therefore RECOMMENDED that measures be taken to guarantee the
integrity and to check the sender's identity of the CID. To that
purpose, one of the counter-measures mentioned above (Section 6.2.2)
SHOULD be used. These measures will either be applied on a packet
level, or globally over the whole CID object. When there is no
packet level integrity verification scheme, it is RECOMMENDED to
digitally sign the CID.

6.3.3. Attacks Against the Object Meta-Data

Corrupting the object meta-data is another way to create a Denial of
Service attack. For example, the attacker changes the MD5 sum
associated to a file. This possibly leads a receiver to reject the
files received, no matter whether the files have been correctly

received or not. When the meta-data are appended to the object,
corrupting the meta-data means that the Compound Object will be
corrupted.

It is therefore RECOMMENDED that measures be taken to guarantee the
integrity and to check the sender's identity of the Compound Object.
To that purpose, one of the counter-measures mentioned above
(Section 6.2.2) SHOULD be used. These measures will either be
applied on a packet level, or globally over the whole Compound
Object. When there is no packet level integrity verification scheme,
it is RECOMMENDED to digitally sign the Compound Object.

6.3.4. Attacks Against the ALC/LCT and NORM Parameters

By corrupting the ALC/LCT header (or header extensions) one can
execute attacks on the underlying ALC/LCT implementation. For
example, sending forged ALC packets with the Close Session flag (A)
set to one can lead the receiver to prematurely close the session.
Similarly, sending forged ALC packets with the Close Object flag (B)
set to one can lead the receiver to prematurely give up the reception
of an object. The same comments can be made for NORM.

It is therefore RECOMMENDED that measures be taken to guarantee the
integrity and to check the sender's identity of each ALC or NORM
packet received. To that purpose, one of the counter-measures
mentioned above (Section 6.2.2) SHOULD be used.

6.3.5. Attacks Against the Associated Building Blocks

Let us first focus on the congestion control building block that may
be used in an ALC or NORM session. A receiver with an incorrect or
corrupted implementation of the multiple rate congestion control
building block may affect the health of the network in the path
between the sender and the receiver. That may also affect the
reception rates of other receivers who joined the session.

When congestion control is applied with FCAST, it is therefore
RECOMMENDED that receivers be required to identify themselves as
legitimate before they receive the Session Description needed to join
the session. If authenticating a receiver does not prevent this
latter to launch an attack, it will enable the network operator to
identify him and to take counter-measures. This authentication can
be made either toward the network operator or the session sender (or
a representative of the sender) in case of NORM. The details of how
it is done are outside the scope of this document.

When congestion control is applied with FCAST, it is also RECOMMENDED
that a packet level authentication scheme be used, as explained in

   Section 6.2.2. Some of them, like TESLA, only provide a delayed
   authentication service, whereas congestion control requires a rapid
   reaction. It is therefore RECOMMENDED [RFC5775] that a receiver
   using TESLA quickly reduces its subscription level when the receiver
   believes that a congestion did occur, even if the packet has not yet
   been authenticated. Therefore TESLA will not prevent DoS attacks
   where an attacker makes the receiver believe that a congestion
   occurred. This is an issue for the receiver, but this will not
   compromise the network. Other authentication methods that do not
   feature this delayed authentication could be preferred, or a group
   MAC scheme could be used in parallel to TESLA to prevent attacks
   launched from outside of the group.

6.4. Other Security Considerations

   Lastly, we note that the security considerations that apply to, and
   are described in, ALC [RFC5775], LCT [RFC5651], NORM [RFC5740] and
   FEC [RFC5052] also apply to FCAST as FCAST builds on those
   specifications. In addition, any security considerations that apply
   to any congestion control building block used in conjunction with
   FCAST also applies to FCAST. Finally, the security discussion of
   [RMT-SEC] also applies here.

6.5. Minimum Security Recommendations

   We now introduce a mandatory to implement but not necessarily to use
   security configuration, in the sense of [RFC3365]. Since FCAST/ALC
   relies on ALC/LCT, it inherits the "baseline secure ALC operation" of
   [RFC5775]. Similarly, since FCAST/NORM relies on NORM, it inherits
   the "baseline secure NORM operation" of [RFC5740]. More precisely,
   in both cases security is achieved by means of IPsec/ESP in transport
   mode. [RFC4303] explains that ESP can be used to potentially provide
   confidentiality, data origin authentication, content integrity, anti-
   replay and (limited) traffic flow confidentiality. [RFC5775]
   specifies that the data origin authentication, content integrity and
   anti-replay services SHALL be used, and that the confidentiality
   service is RECOMMENDED. If a short lived session MAY rely on manual
   keying, it is also RECOMMENDED that an automated key management
   scheme be used, especially in case of long lived sessions.

   Therefore, the RECOMMENDED solution for FCAST provides per-packet
   security, with data origin authentication, integrity verification and
   anti-replay. This is sufficient to prevent most of the in-band
   attacks listed above. If confidentiality is required, a per-packet
   encryption SHOULD also be used.

7. IANA Considerations

7.1. Namespace declaration for Object Meta-Data Format

   This document requires a IANA registration for the following name-
   space: "Object Meta-Data Format" (MDFmt). Values in this namespace
   are 4-bit positive integers between 0 and 15 inclusive and they
   define the format of the object meta-data ((see Section 5.1).

   Initial values for the LCT Header Extension Type registry are defined
   in Section 7.1.1. Future assignments are to be made through Expert
   Review [RFC5226].

7.1.1. Object Meta-Data Format registration

   This document registers one value in the "Object Meta-Data Format"
   namespace as follows:

```
        +----------------------------------------+-------------+
        |               format name              |    Value    |
        +----------------------------------------+-------------+
        | as per HTTP/1.1 metainformation format | 0 (default) |
        +----------------------------------------+-------------+
```

   All implementations MUST support format 0 (default).

7.2. Namespace declaration for Object Meta-Data Encoding

   This document requires a IANA registration for the following name-
   space: "Object Meta-Data Encoding" (MDEnc). Values in this namespace
   are 4-bit positive integers between 0 and 15 inclusive and they
   define the optional encoding of the Object Meta-Data field (see
   Section 5.1).

   Initial values for the LCT Header Extension Type registry are defined
   in Section 7.2.1. Future assignments are to be made through Expert
   Review [RFC5226].

7.2.1. Object Meta-Data Encoding registration

   This document registers two values in the "Object Meta-Data Encoding"
   namespace as follows:

```
+------------+-------------+
|    Name    |    Value    |
+------------+-------------+
| plain text | 0 (default) |
|    gzip    |      1      |
+------------+-------------+
```

All implementations MUST support both value 0 (plain-text, default)
and value 1 (gzip).

8. Acknowledgments

The authors are grateful to the authors of [ALC-00] for specifying
the first version of FCAST/ALC. The authors are also grateful to
Gorry Fairhurst and Lorenzo Vicisano for their valuable comments.

9. References

9.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5226]   Narten, T. and H. Alvestrand, "Guidelines for Writing an
            IANA Considerations Section in RFCs", BCP 26, RFC 5226,
            May 2008.

[RFC1071]   Braden, R., Borman, D., Partridge, C., and W. Plummer,
            "Computing the Internet checksum", RFC 1071,
            September 1988.

[RFC5651]   Luby, M., Watson, M., and L. Vicisano, "Layered Coding
            Transport (LCT) Building Block", RFC 5651, October 2009.

[RFC5740]   Adamson, B., Bormann, C., Handley, M., and J. Macker,
            "NACK-Oriented Reliable Multicast (NORM) Transport
            Protocol", RFC 5740, November 2009.

[RFC5775]   Luby, M., Watson, M., and L. Vicisano, "Asynchronous
            Layered Coding (ALC) Protocol Instantiation", RFC 5775,
            April 2010.

[RFC1952]   Deutsch, P., Gailly, J-L., Adler, M., Deutsch, L., and G.
            Randers-Pehrson, "GZIP file format specification version
            4.3", RFC 1952, May 1996.

   [RFC2616]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
                Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
                Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

9.2.   Informative References

   [ALC-00]   Luby, M., Gemmell, G., Vicisano, L., Crowcroft, J., and B.
                Lueckenhoff, "Asynchronous Layered Coding: a Scalable
                Reliable Multicast Protocol", March 2000.

   [RMT-FLUTE]
                Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen,
                "FLUTE - File Delivery over Unidirectional Transport",
                Work in Progress, February 2011.

   [RFC3365]   Schiller, J., "Strong Security Requirements for Internet
                Engineering Task Force Standard Protocols", BCP 61,
                RFC 3365, August 2002.

   [RMT-SEC]   Roca, V., Adamson, B., and H. Asaeda, "Security and
                Reliable Multicast Transport Protocols: Discussions and
                Guidelines", Work in progress,
                 draft-ietf-rmt-sec-discussion-06.txt, March 2011.

   [RFC2104]   Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-
                Hashing for Message Authentication", RFC 2104,
                February 1997.

   [RFC3447]   Jonsson, J. and B. Kaliski, "Public-Key Cryptography
                Standards (PKCS) #1: RSA Cryptography Specifications
                Version 2.1", RFC 3447, February 2003.

   [RFC4082]   Perrig, A., Song, D., Canetti, R., Tygar, J., and B.
                Briscoe, "Timed Efficient Stream Loss-Tolerant
                Authentication (TESLA): Multicast Source Authentication
                Transform Introduction", RFC 4082, June 2005.

   [RFC4303]   Kent, S., "IP Encapsulating Security Payload (ESP)",
                RFC 4303, December 2005.

   [RFC5052]   Watson, M., Luby, M., and L. Vicisano, "Forward Error
                Correction (FEC) Building Block", RFC 5052, August 2007.

   [RFC5510]   Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo,
                "Reed-Solomon Forward Error Correction (FEC) Schemes",
                RFC 5510, April 2009.

   [RFC5776]   Roca, V., Francillon, A., and S. Faurite, "Use of Timed

            Efficient Stream Loss-Tolerant Authentication (TESLA) in
            the Asynchronous Layered Coding (ALC) and NACK-Oriented
            Reliable Multicast (NORM) Protocols", RFC 5776,
            April 2010.

   [RMT-SIMPLE-AUTH]
            Roca, V., "Simple Authentication Schemes for the ALC and
            NORM Protocols", Work in
            progress draft-ietf-rmt-simple-auth-for-alc-norm-04.txt,
            July 2011.

Appendix A. FCAST Examples

A.1. Basic Examples

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0 |   0   |1|0|   0   |   0   |              Checksum          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              44                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
.                                                               .
.        meta-data ASCII null terminated string (33 bytes)      .
.                                                               .
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               |                  Padding                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                         Object data                           .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
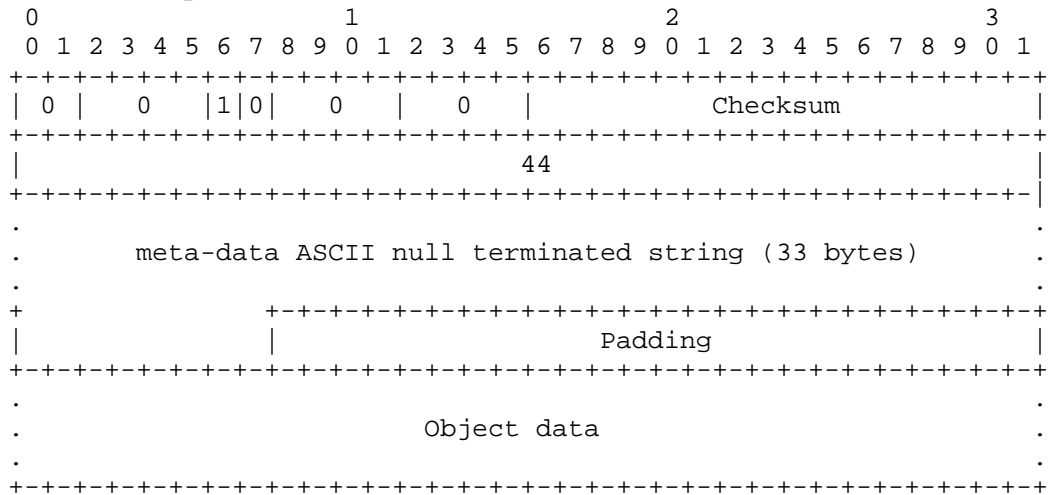
                   Figure 4: Compound Object Example.

   Figure 4 shows a regular Compound Object where the meta-data ASCII
   string, in HTTP/1.1 meta-information format (MDFmt=0) contains:

   Content-Location: example.txt <CR-LF>

   This string is 33 bytes long, including the NULL-termination
   character. There is no gzip encoding of the meta-data (MDEnc=0) and
   there is no Content-Length information either since this length can
   easily be calculated by the receiver as the FEC OTI transfer length
   minus the header length. Finally, the checksum encompasses the whole
   Compound Object (G=1).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0 |   0   |1|0|   0   |   0   |            Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               8                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                         Object data                          .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
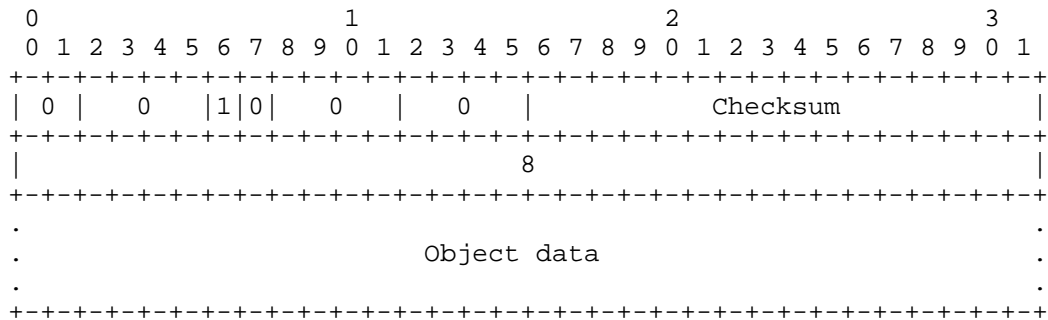
           Figure 5: Compound Object Example with no Meta-Data.

Figure 5 shows a Compound Object without any meta-data. The fact
there is no meta-data is indicated by the value 8 of the Compound
Object Header Length field. No padding is required.

Figure 6 shows an example CID object, in the case of a static FCAST
session, i.e., a session where the set of objects is set once and for
all. There is no meta-data in this example since Fcast-CID-Complete
and Fcast-CID-ID are both implicit.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0 |   0   |1|1|   0   |   0   |            Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               8                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
.                                                               .
.                      Object List string                      .
.                                                               .
.                              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
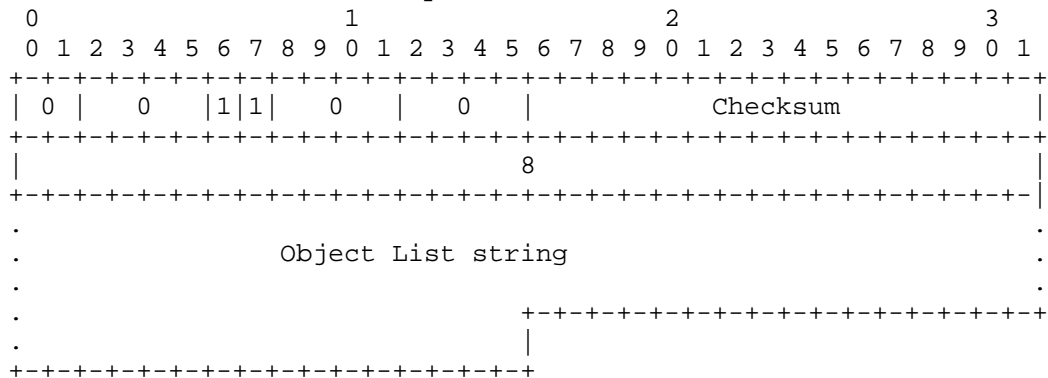
         Figure 6: Example of CID, in case of a static session.

The object list contains the following 26 byte long string, including
the NULL-termination character:

1,2,3,100-104,200-203,299

There are therefore a total of 3+5+4+1 = 13 objects in the carousel
instance, and therefore in the FCAST session. There is no meta-data
associated to this CID. The session being static and composed of a
single Carousel Instance, the sender did not feel the necessity to
carry a Carousel Instance ID meta-data.

A.2. FCAST/NORM with NORM_INFO Examples

   In case of FCAST/NORM, the FCAST Compound Object meta-data (or a
   subset of it) can be carried as part of a NORM_INFO message, as a new
   Compound Object that does not contain any Compound Object Data. In
   the following example we assume that the whole meta-data is carried
   in such a message for a certain Compound Object. Figure 7 shows an
   example NORM_INFO message that contains the FCAST Compound Object
   Header and meta-data as its payload. In this example, the first 16
   bytes are the NORM_INFO base header, the next 12 bytes are a NORM
   EXT_FTI header extension containing the FEC Object Transport
   Information for the associated object, and the remaining bytes are
   the FCAST Compound Object Header and meta-data. Note that "padding"
   MUST NOT be used and that the FCAST checksum only encompasses the
   Compound Object Header (G=0).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ --
 |version| type=1|  hdr_len = 7  |            sequence           |  ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |                           source_id                           |  n
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  o
 |         instance_id           |     grtt      |backoff| gsize |  r
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  m
 |     flags     |  fec_id = 5   |     object_transport_id       |  v
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ --
 |    HET = 64   |    HEL = 3     |                               |  ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +  f
 |                     Transfer Length = 41                      |  t
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  i
 |   Encoding Symbol Length (E)  |  MaxBlkLen (B) |    max_n      |  v
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ --
 | 0 |   0   |0|0|  0   |   0    |            Checksum            |  ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |                              41                               |  f
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-| c
 .                                                             .  a
 .      meta-data ASCII null terminated string (33 bytes)      .  s
 .                                                             .  t
 +              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |              |                                                 v
 +-+-+-+-+-+-+-+-+                                              --
```

           Figure 7: NORM_INFO containing an EXT_FTI header extension and an
                      FCAST Compound Object Header

   The NORM_INFO message shown in Figure 7 contains the EXT_FTI header
   extension to carry the FEC OTI. In this example, the FEC OTI format

is that of the Reed-Solomon FEC coding scheme for fec_id = 5 as
described in [RFC5510]. Other alternatives for providing the FEC OTI
would have been to either include it directly in the meta-data of the
FCAST Compound Header, or to include an EXT_FTI header extension to
all NORM_DATA packets (or a subset of them). Note that the NORM
"Transfer_Length" is the total length of the associated FCAST
Compound Object, i.e., 41 bytes.

The FCAST Compound Object in this example does contain the same meta-
data and is formatted as in the example of Figure 4. With the
combination of the FEC_OTI and the FCAST meta-data, the NORM protocol
and FCAST application have all of the information needed to reliably
receive and process the associated object. Indeed, the NORM protocol
provides rapid (NORM_INFO has precedence over the associated object
content), reliable delivery of the NORM_INFO message and its payload,
the FCAST Compound Object Header.

Authors' Addresses

   Vincent Roca
   INRIA
   655, av. de l'Europe
   Inovallee; Montbonnot
   ST ISMIER cedex 38334
   France

   Email: vincent.roca@inria.fr
   URI: http://planete.inrialpes.fr/people/roca/

   Brian Adamson
   Naval Research Laboratory
   Washington, DC 20375
   USA

   Email: adamson@itd.nrl.navy.mil
   URI: http://cs.itd.nrl.navy.mil